



Escola Politécnica da USP - Depto. de Enga. Mecatrônica

PMR-3510 Inteligência Artificial

AULA 14 - AI PLANNING ALGORITHMS

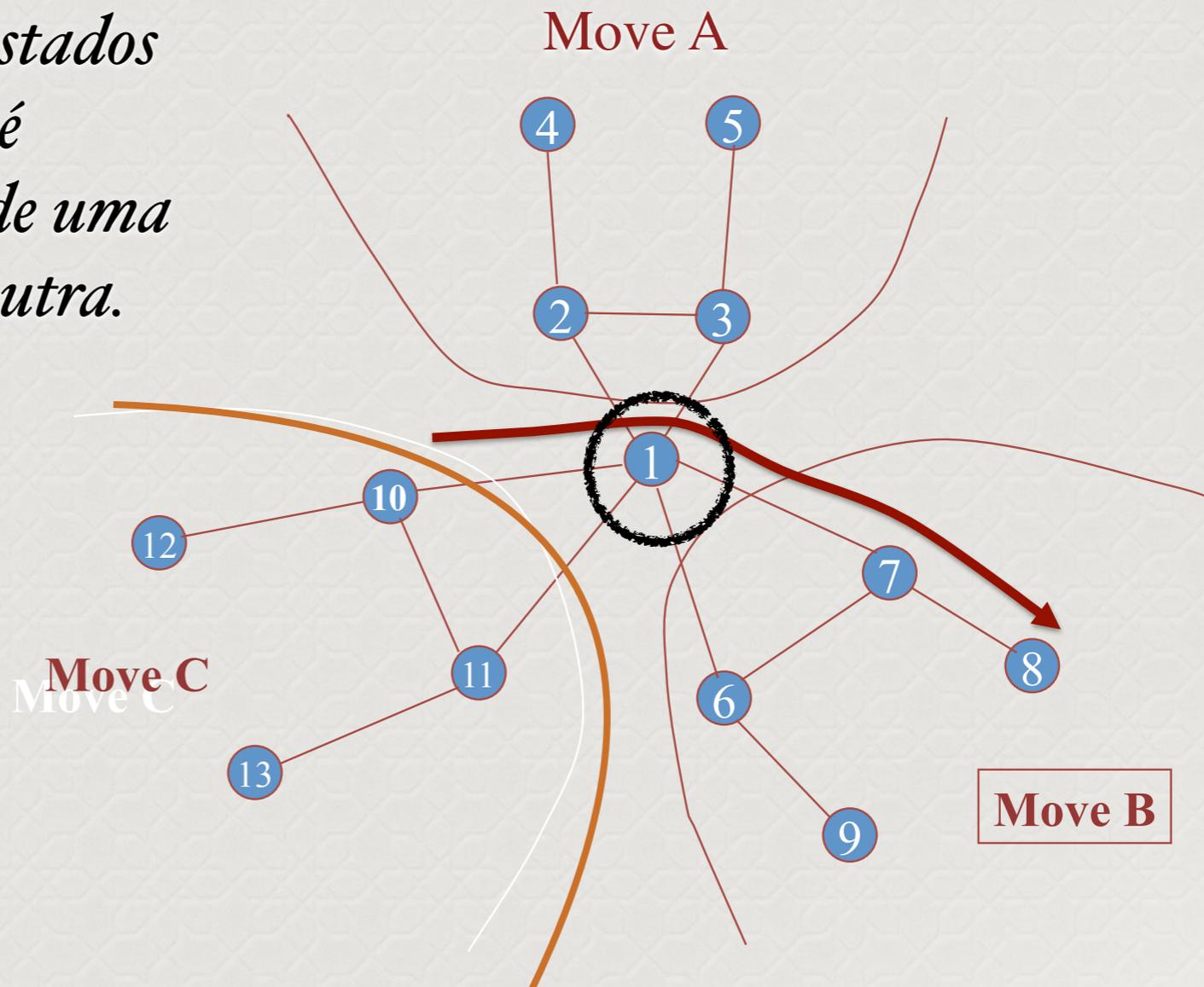
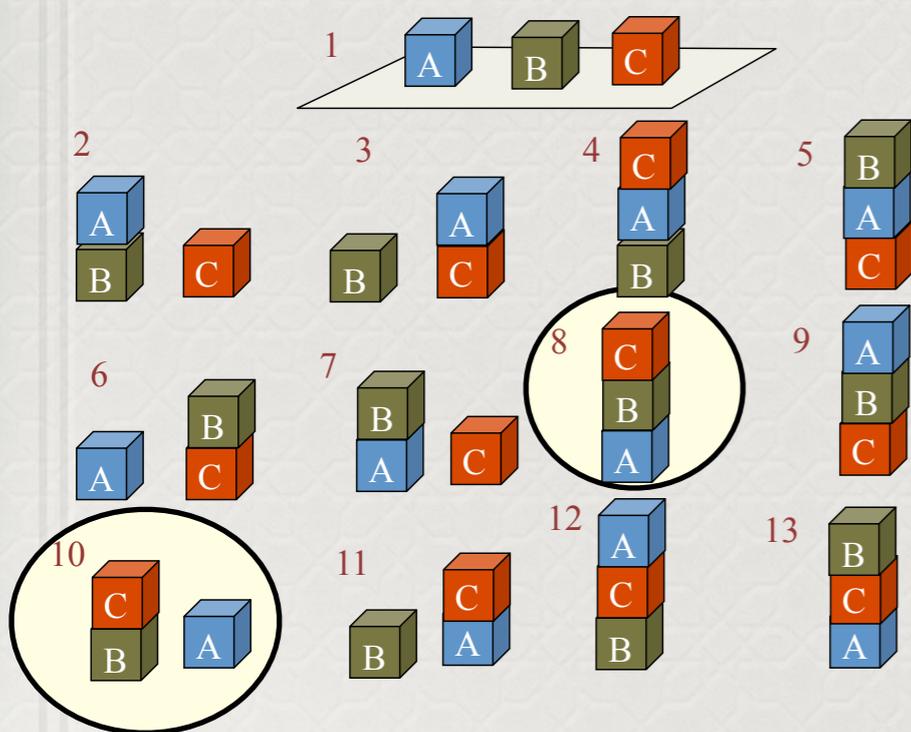
Prof. José Reinaldo Silva

reinaldo@usp.br



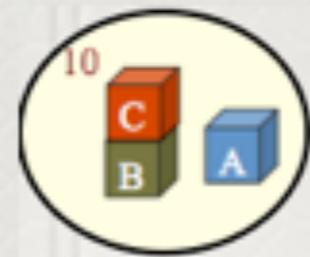


Uma análise do espaço de estados mostra que o estado 1 é "preferencial" para passar de uma componente conectada a outra.





Representação dos "estados" em lógica:
estado inicial e estado "preferencial.



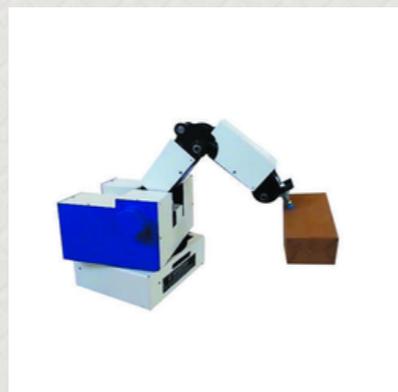
sobre(c, b)
sobre(b, mesa)
sobre(a, mesa)
livre(c)
livre(a)

sobre(c, mesa)
sobre(b, mesa)
sobre(a, mesa)
livre(c)
livre(b)
livre(a)



estado inicial estado objetivo

Conjunto de Ações



Problema de planejamento

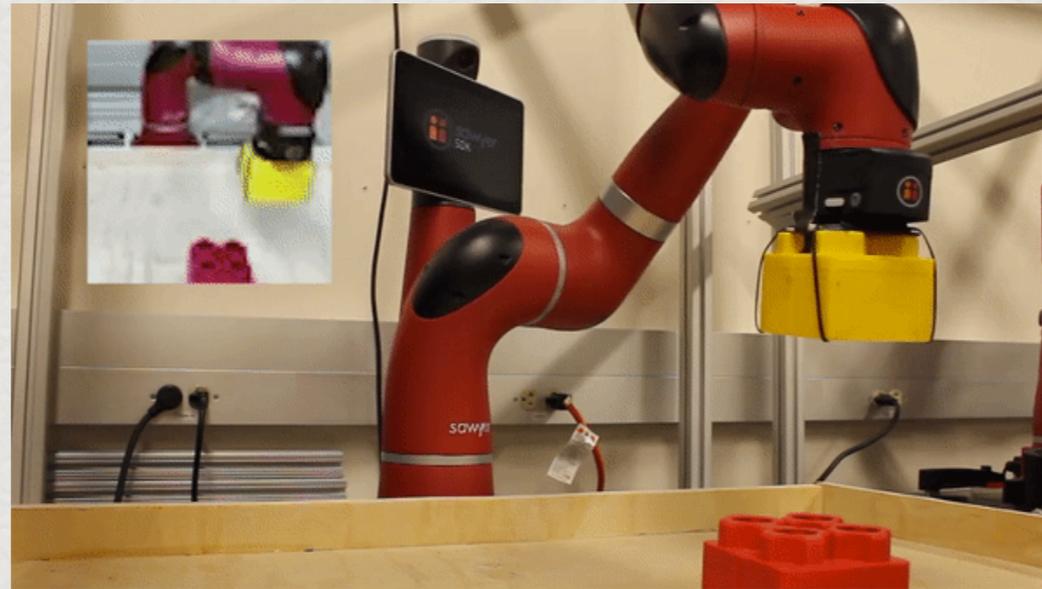
Critério de aplicação das ações

Percepção das mudanças de estado

Análise de custo

Base de conhecimento

O conjunto de ações é baseado nos movimentos de um braço robótico onde o efetuator (garra) só pode segurar um bloco de cada vez. A deposição só pode ser feita sobre um bloco livre, que não tenha nenhum outro bloco em cima dele.



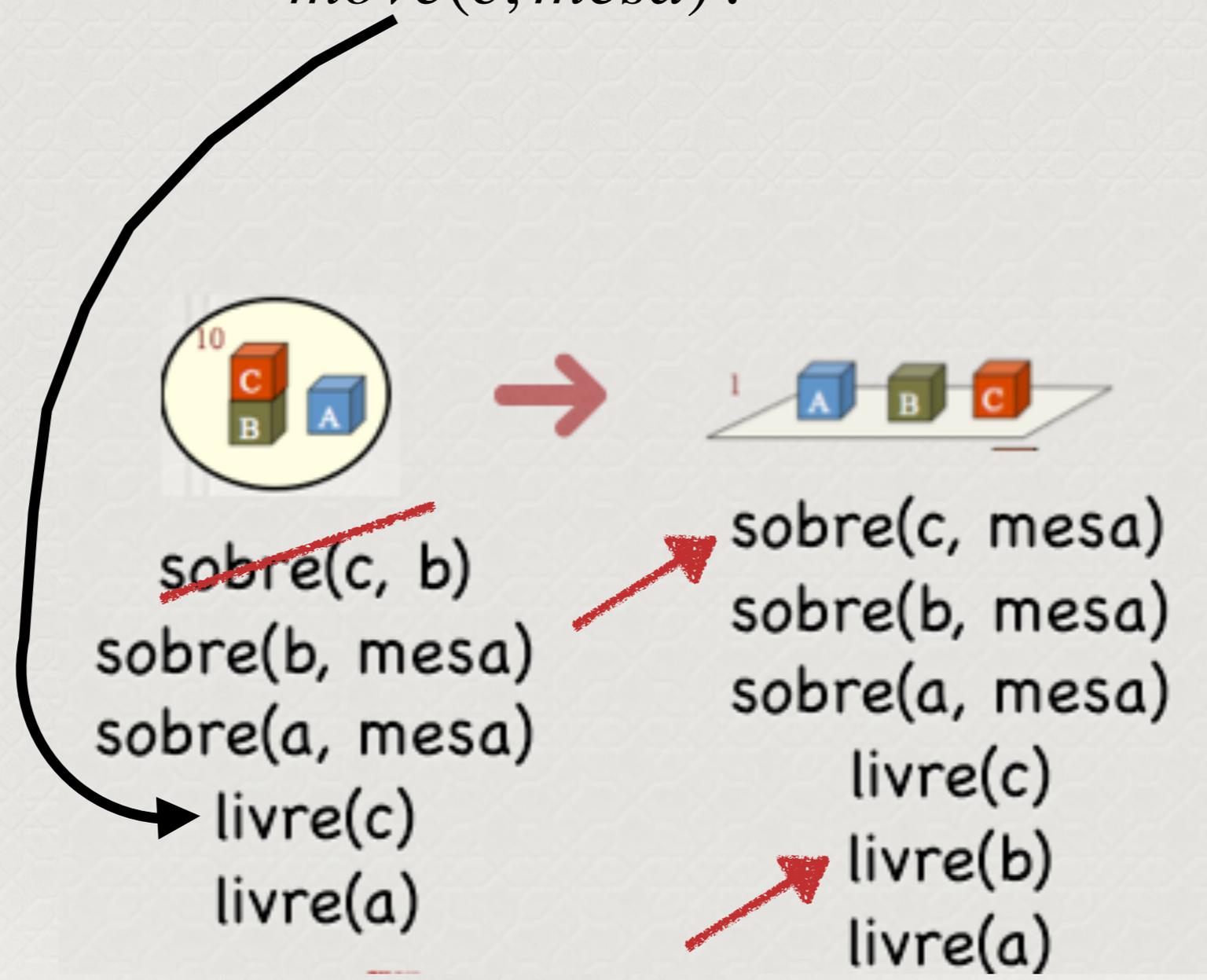
uma ação desse tipo pode ser definida como $move(X, Y)$, onde um robô pode mover o bloco X para cima do bloco Y ou para a mesa.



Pré-condições

Pós-condições

$move(c, mesa)$.

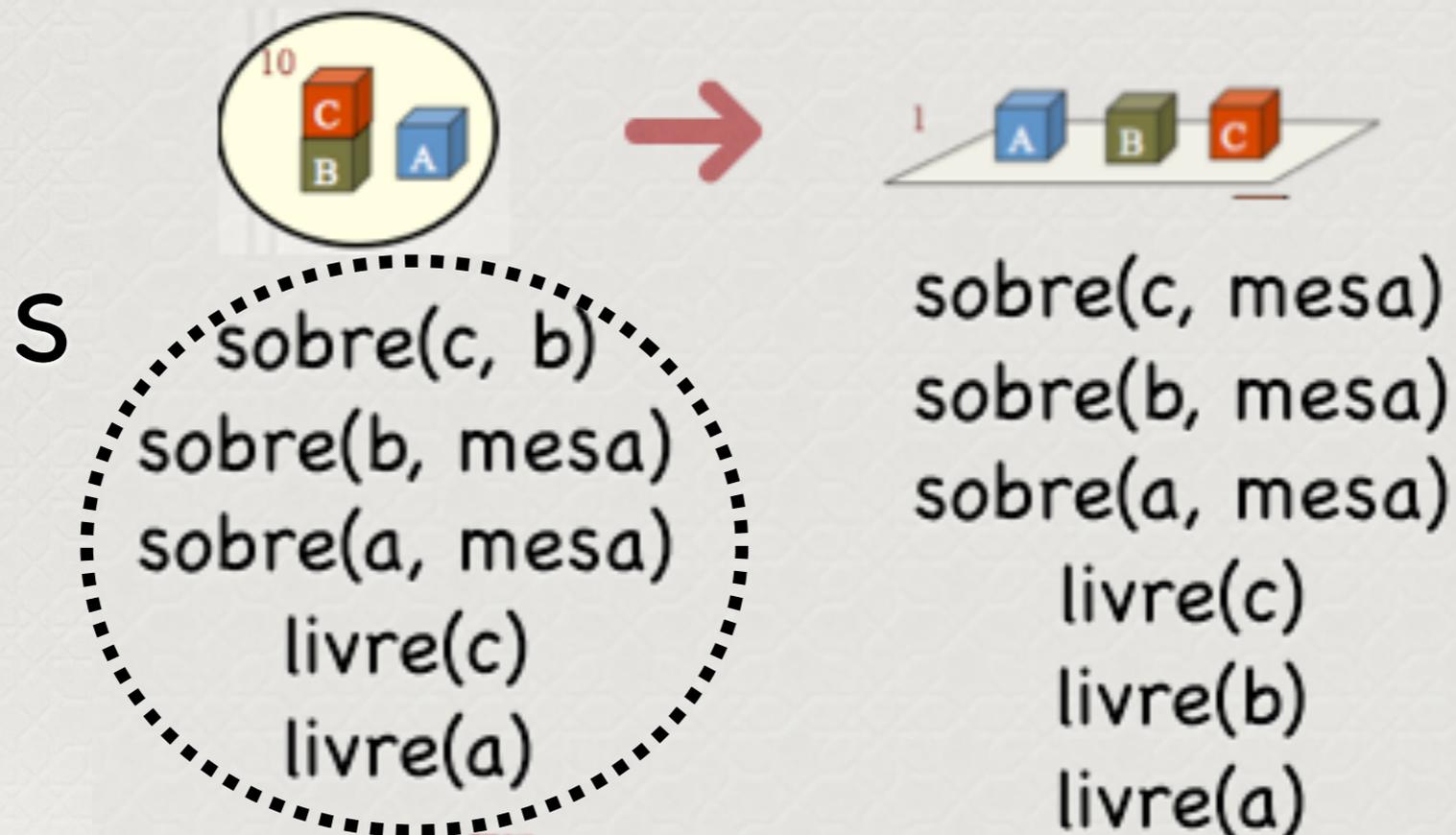


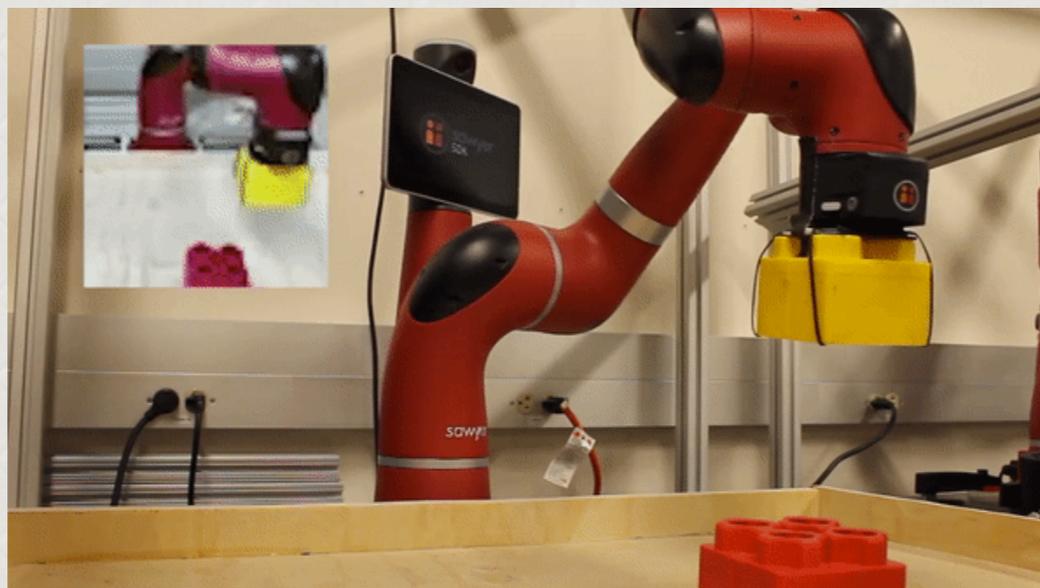


move(c, mesa).

remove([sobre(c,b)], s)

add([sobre(c,mesa), livre(b)], s)





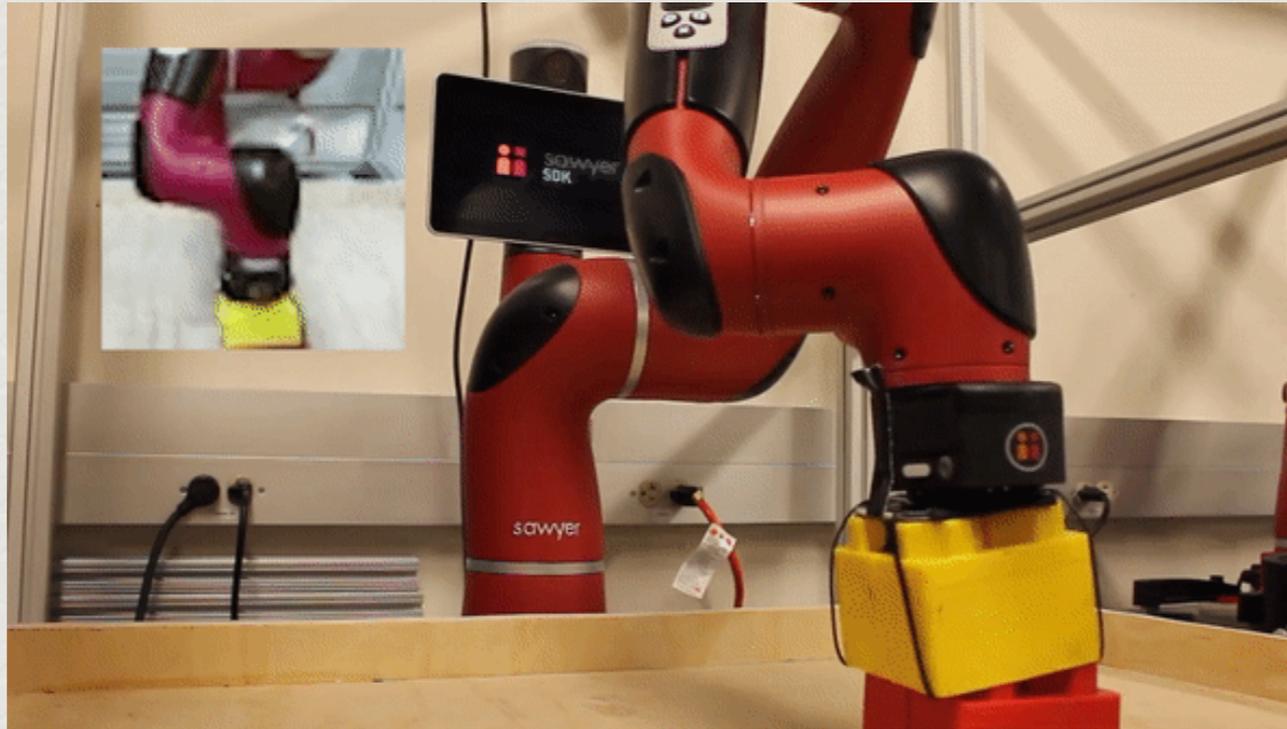
move(X, Y) : – livre(X), livre(Y), livre(garra), remove(livre(Y)), add(sobre(X, Y)).



Intuitivamente, as ações não são somente enumeradas mas definidas como predicados, com regras para sua aplicação. O evento é aplicado identificando uma lista de termos que deve ser removida da descrição do estado origem, e acrescentando uma nova lista a esse estado para obter o estado destino.



Será que isso pode ser generalizado como um método para resolver uma classe ampla de problemas de planejamento?



vamos discutir essa possibilidade e apresentar
mais detalhadamente um algoritmo genérico
para resolver problemas conhecido como
STRIPS.



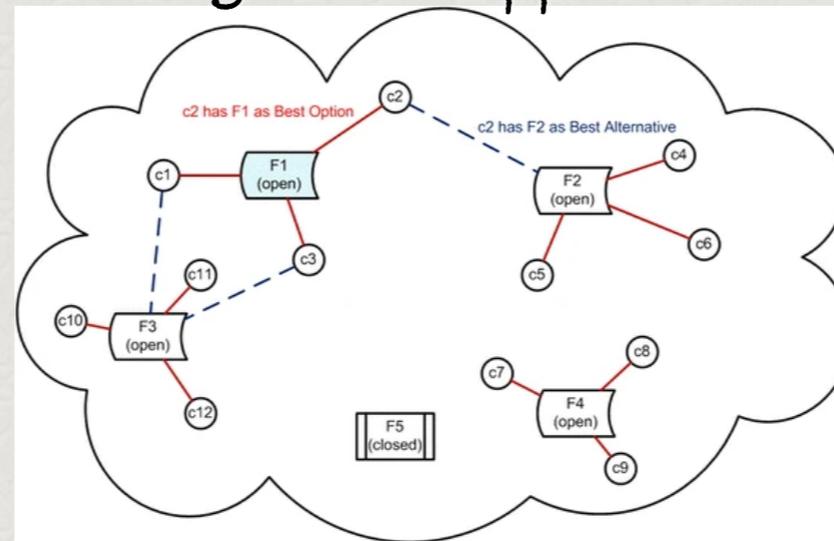
vamos também começar a tratar dos aspectos práticos aplicados ao problema-modelo do mundo de blocos, que será o próximo exercício-programa.



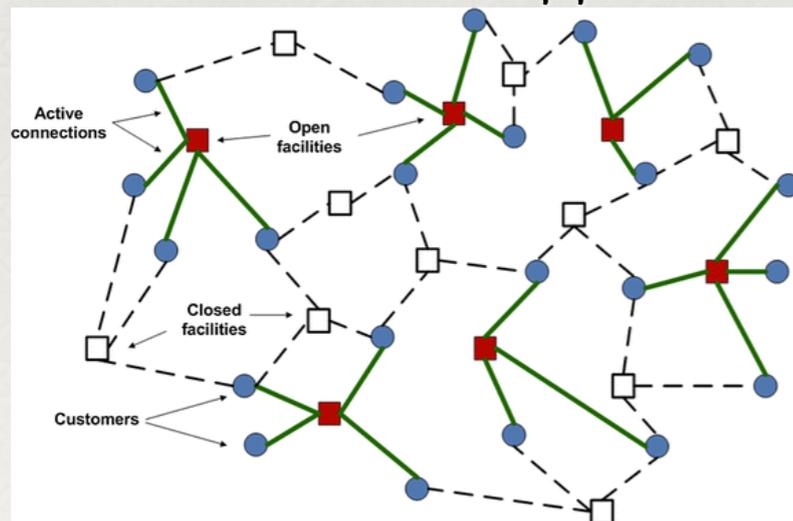


General Problem-solving approach

Cognitive approach



Deterministic approach



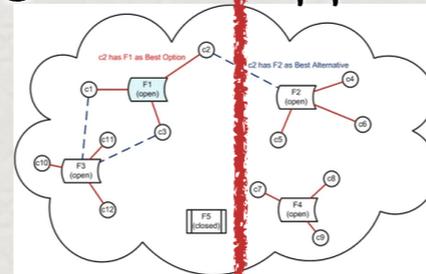
Stochastic and uncertainty approach





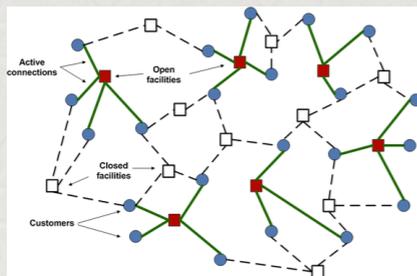
General Problem-solving approach

Cognitive approach



Modelos discretos, com heurísticas ou baseados em conhecimento (IA clássica).

Deterministic approach

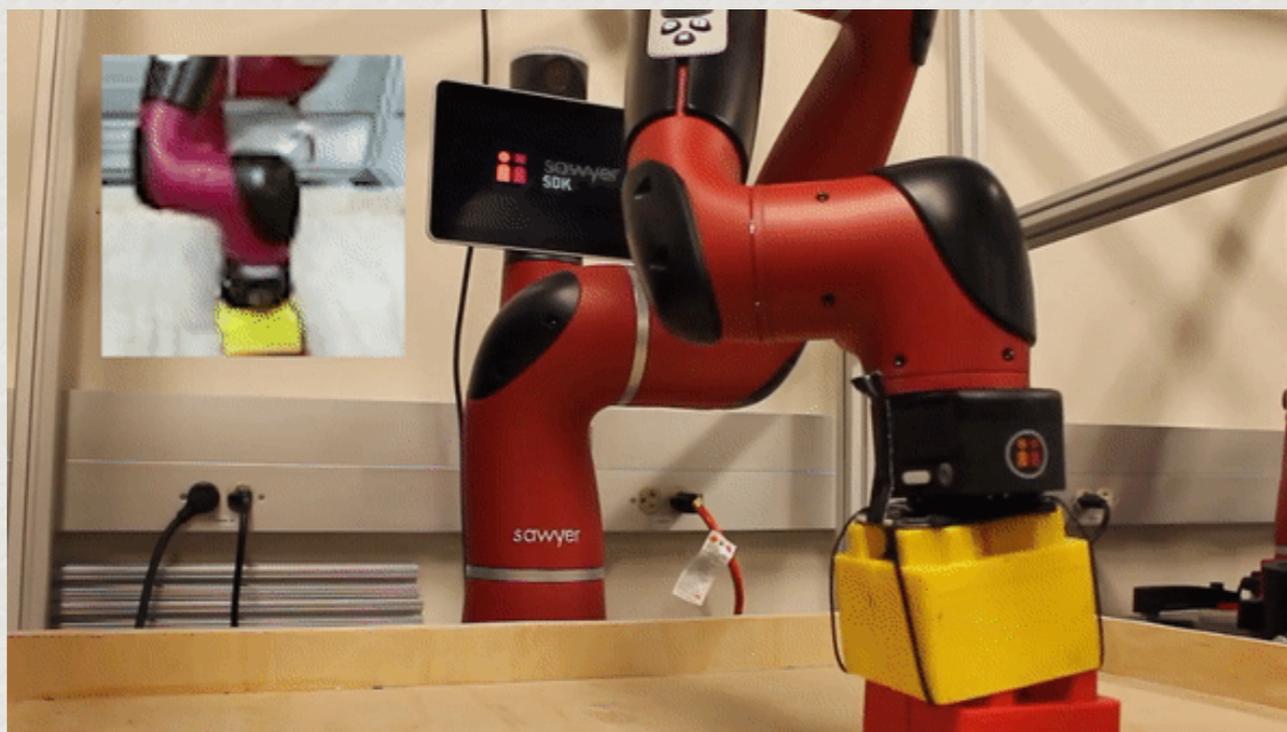


Modelos contínuos, discretos, computacionais.

Stochastic and uncertainty approach



Modelos discretos, estocásticos, baseados e bigdata.



Continuaremos a discussão sobre a proposta do STRIPS, e chegaremos até um algoritmo genérico implementado sobre o mundo de blocos.

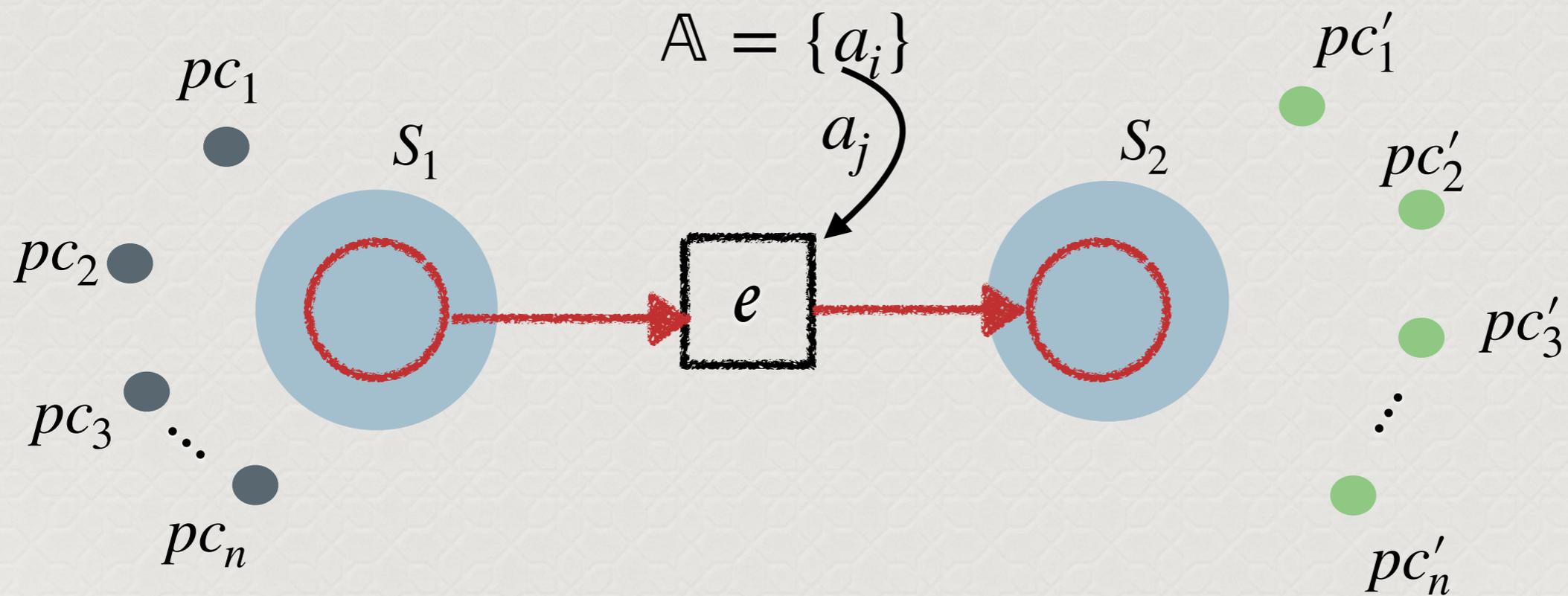


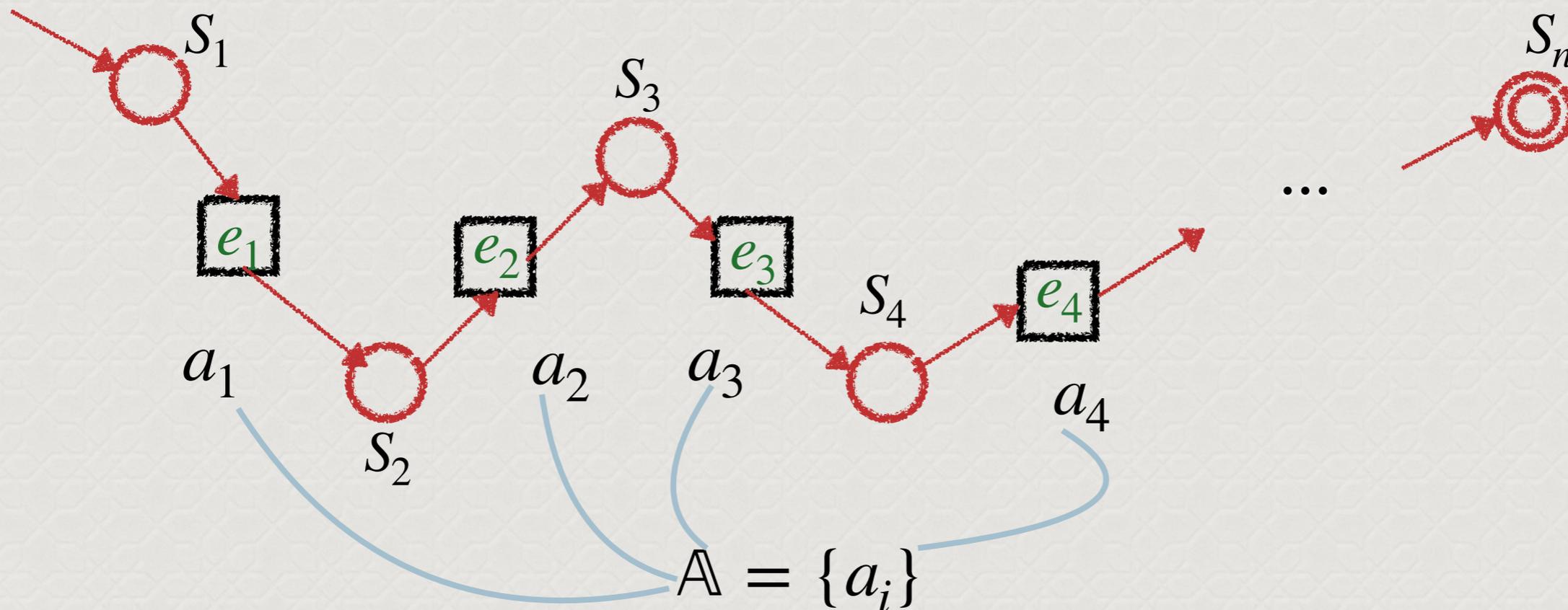
STATE TRANSITION DIAGRAM: NOTATION



O planejamento clássico é fundamentado no paradigma estado-transição, assumindo que o “plano” será executado de forma discreta, uma ação por vez.

A cada instante, o sistema estará em um estado específico que “admite” ações que podem fazê-lo evoluir para um próximo estado.





11.1 Definition of Classical Planning

Classical planning

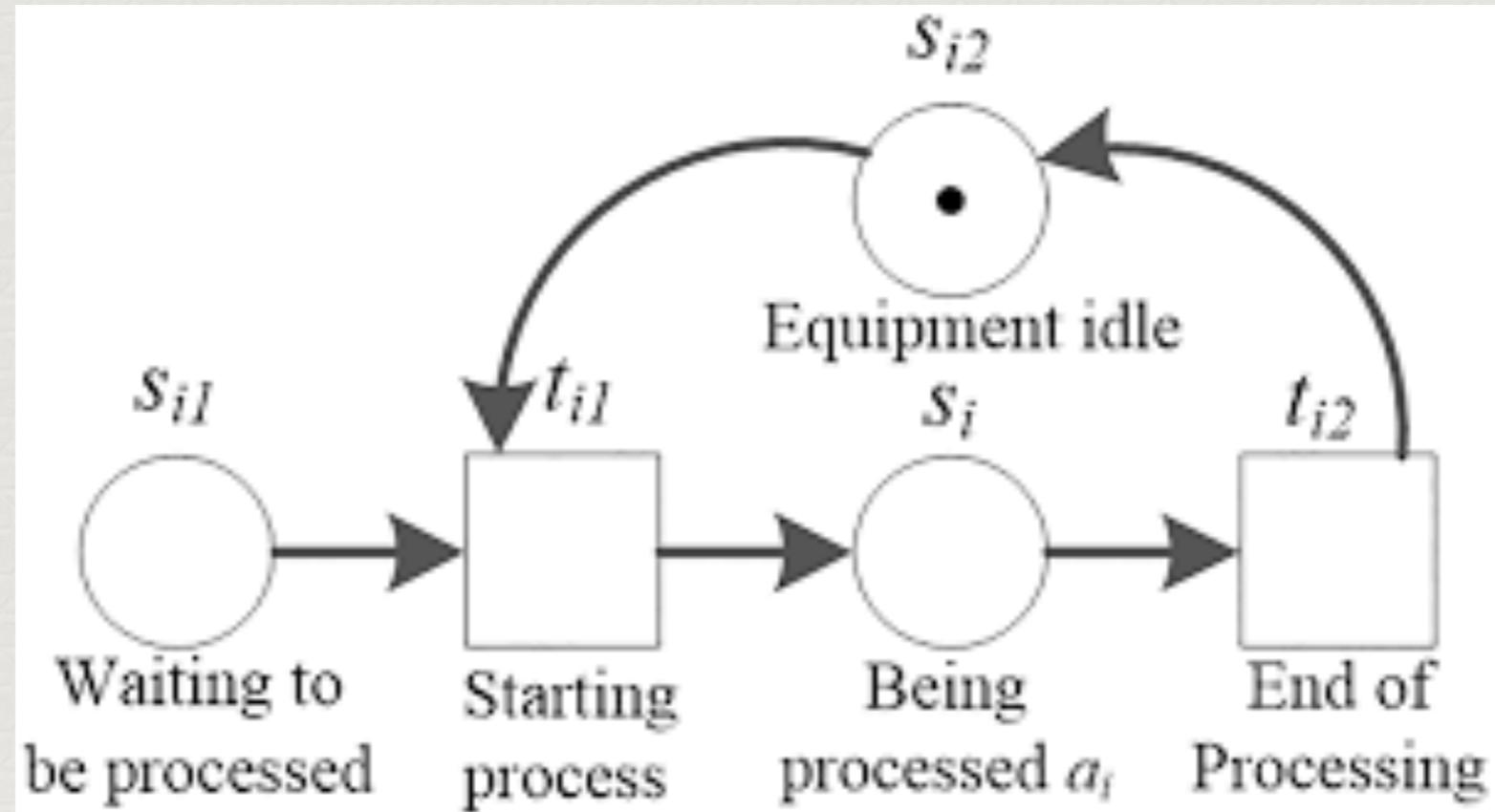
Classical planning is defined as the task of finding a sequence of actions to accomplish a goal in a discrete, deterministic, static, fully observable environment. We have seen two approaches to this task: the problem-solving agent of Chapter 3 and the hybrid propositional logical agent of Chapter 7. Both share two limitations. First, they both require ad hoc heuristics for each new domain: a heuristic evaluation function for search, and hand-written code for the hybrid wumpus agent. Second, they both need to explicitly represent an exponentially large state space. For example, in the propositional logic model of the wumpus world, the axiom for moving a step forward had to be repeated for all four agent orientations, T time steps, and n^2 current locations.



AI Magazine Volume 11 Number 2 (1990) (© AAAI)

AI Planning: Systems and Techniques¹

James Hendler, Austin Tate, and Mark Drummond





General Planning approach

Domain independent



Domain dependent





O planejamento clássico é basicamente um processo de achar um arranjo de k ações pertencentes a um conjunto dado que leve o sistema do estado inicial ao estado final fornecido no "planning problem". Se todas as sequências de ações (sem repetições) fossem possíveis teríamos algo como $k!$ possibilidades. Contando com as repetições e com o fato de que nem todas as sequências são válidas fisicamente, ainda assim o número de sequências cresce exponencialmente com o número de estados.



O método geral de resolução consiste em identificar como se configura estado (no caso por uma lista de predicados e constantes) e como se faz a “transição”, modificando esta configuração de forma discreta (subtraindo alguns predicados e acrescentando outros).

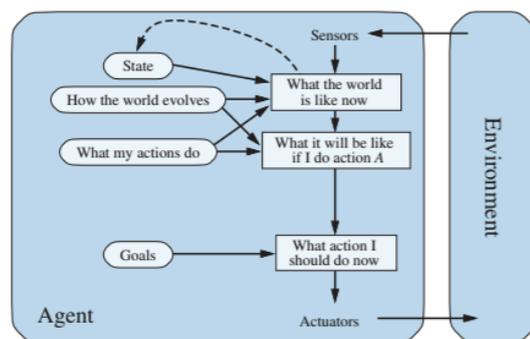


Figure 2.13 A model-based, goal-based agent. It keeps track of the world state as well as a set of goals it is trying to achieve, and chooses an action that will (eventually) lead to the achievement of its goals.



Heurística ou regras de inferência



Existe um métodos geral, independente de domínio, para resolver problemas, em especial o problema de agentes com objetivo?.





STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving¹

Richard E. Fikes

Nils J. Nilsson

Stanford Research Institute, Menlo Park, California

Recommended by B. Raphael

Presented at the 2nd IJCAI, Imperial College, London, England, September 1-3, 1971.

ABSTRACT

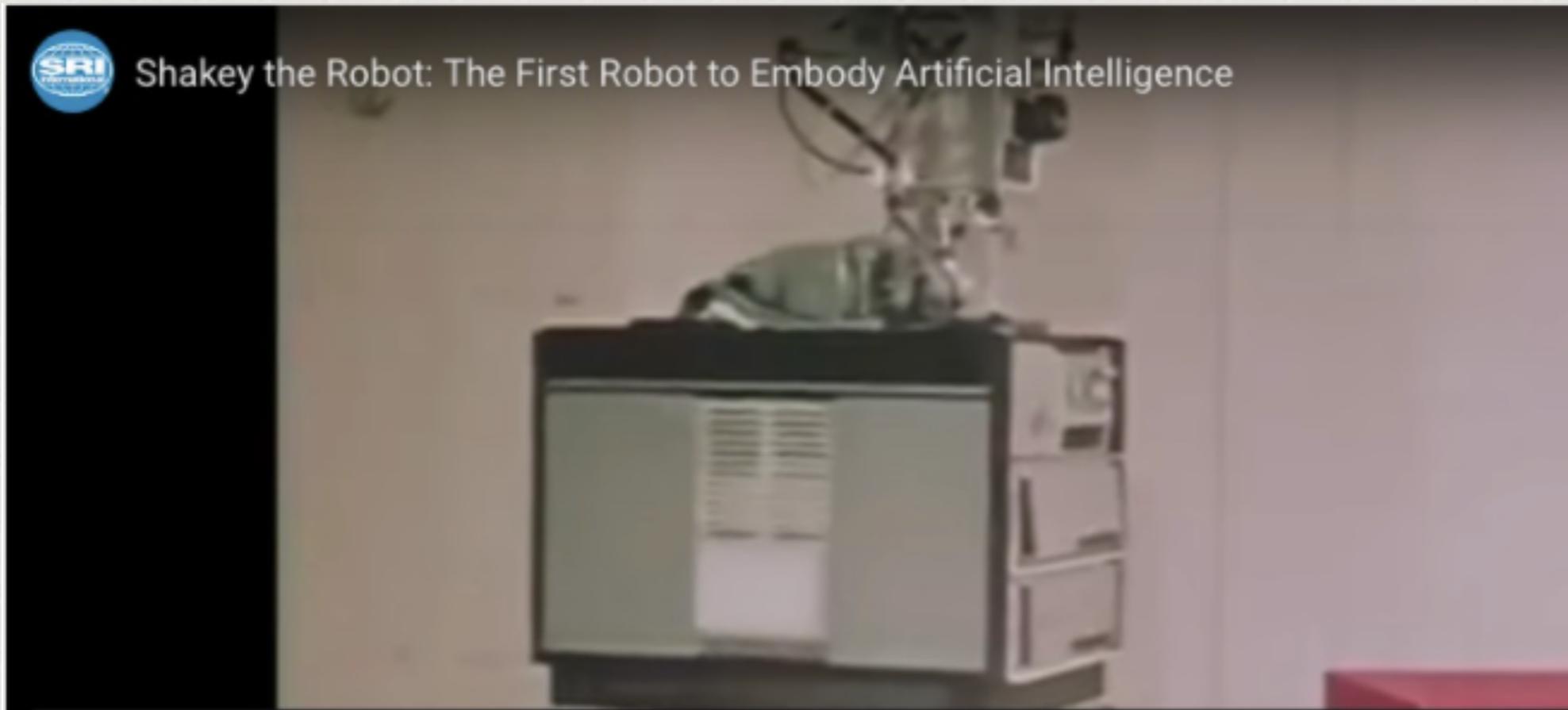
We describe a new problem solver called STRIPS that attempts to find a sequence of operators in a space of world models to transform a given initial world model into a model in which a given goal formula can be proven to be true. STRIPS represents a world model as an arbitrary collection of first-order predicate calculus formulas and is designed to work with models consisting of large numbers of formulas. It employs a resolution theorem prover to answer questions of particular models and uses means-ends analysis to guide it to the desired goal-satisfying model.

DESCRIPTIVE TERMS

Problem solving, theorem proving, robot planning, heuristic search.

1. Introduction

This paper describes a new problem-solving program called STRIPS (Stanford Research Institute Problem Solver). An initial version of the program has been implemented in LISP on a PDP-10 and is being used in conjunction with robot research at SRI. STRIPS is a member of the class of





Graph Search and STRIPS Planning

1 Introduction to Graph Search

Consider the eight puzzle show in figure 1. This puzzle is played on a three by three square board containing eight tiles and an empty square. Any tile which is next to the empty square can be slid into the empty square leaving an opening (empty square) in the place that used to be occupied by the moved tile. Figure 1 shows how one state of the puzzle can be transformed into another configure by sliding a tile into the empty square. Given an initial state of the puzzle the objective is to find a sequence of legal moves that transform the initial state into some desired goal state. Figure 2 also shows a typical goal state.

The problem of finding a sequence of moves that transforms a given initial state into a given goal state can be formulated as a graph search problem. The nodes of the graph are the possible states of the puzzle and the arcs of the graph correspond to legal moves that transform one state into another. The problem is to find a path in this graph from a given initial state to a given goal state.

In the eight puzzle there are four types of legal moves. The empty square can move up, move down, move right, or move left. This allows the puzzle to be formulated in terms of four operations U , D , L , and R that move the empty square up, down, right and left respectively. If the empty square is already on the upper edge of the board then we define the operation U to



Definition: A *STRIPS* planning problem consists of a STRIPS operator specification, a set Σ of initial propositions and a set Ω of goal propositions.

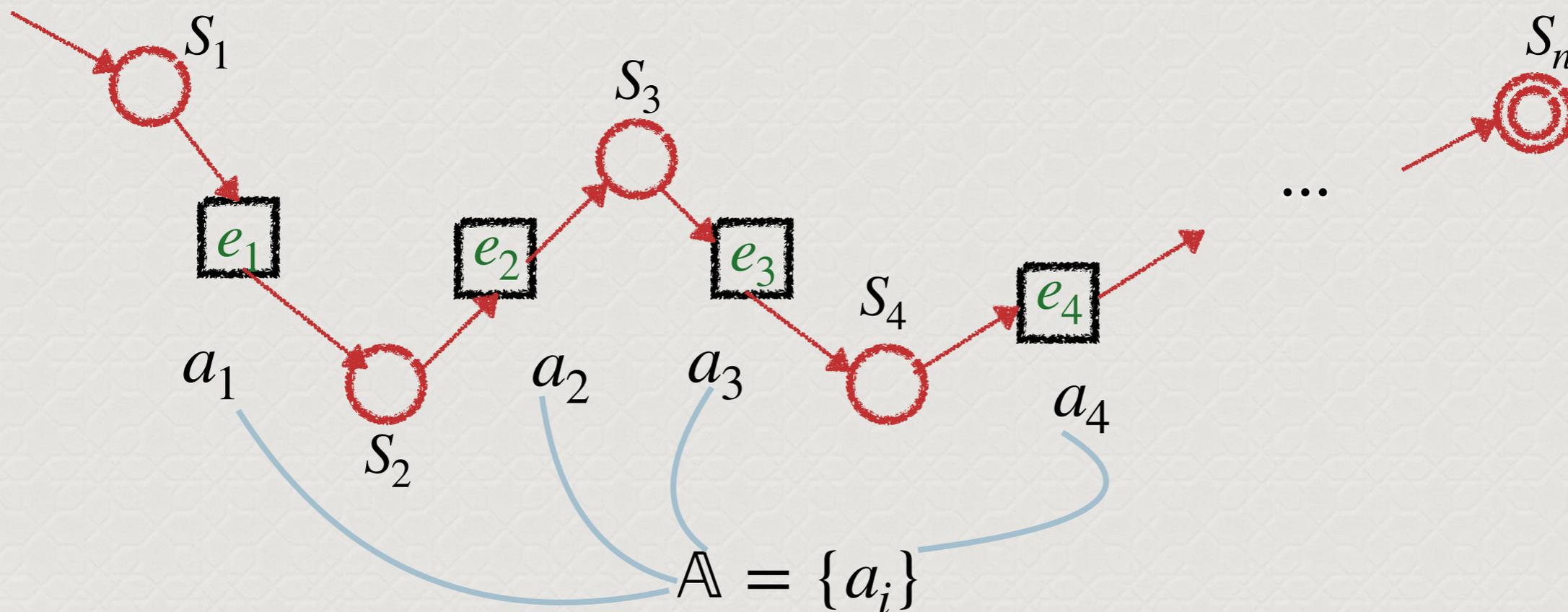
Definition: A *solution* to a STRIPS planning problem is a plan (sequence of operators) α such that, in every graph where the STRIPS operator specification holds, we have $\Sigma \rightarrow [\alpha]\Omega$.



Definition: A *STRIPS operator specification* consists of a set of *operator symbols* where each operator symbol is associated with a *prerequisite list*, an *add list* and a *delete list* each of which is a set of proposition symbols.

Definition: A STRIPS operator specification is said to *hold* (or be *valid*) in a graph search problem if for each operator o_i , and each node n such that every prerequisite of o_i is true at n , we have the following conditions.

- All propositions on the add list of o_i are true at the node $o_i(n)$.
- If P is a proposition that is *not* on the delete list of o_i , and P is true at n , then P is true at $o_i(n)$.



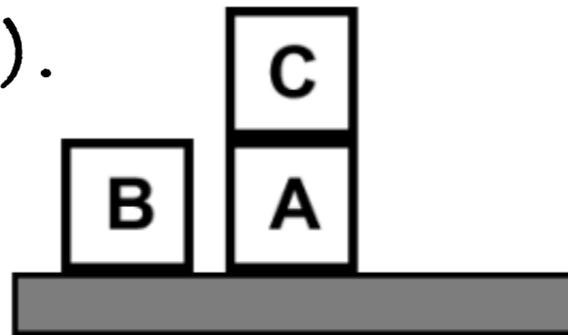
- Apesar de geral o algoritmo não é completo, e pode gerar a anomalia de Sussman;
- Não há nenhuma garantia de convergência, portanto, a busca de solução pode divergir;
- O algoritmo não é otimizante.



Sussman Anomaly in the block world

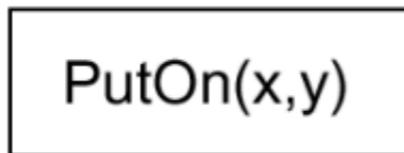
"Sussman anomaly" problem

on(b, table).
on(a, table).
on(c, a).
free(b).
free(c).

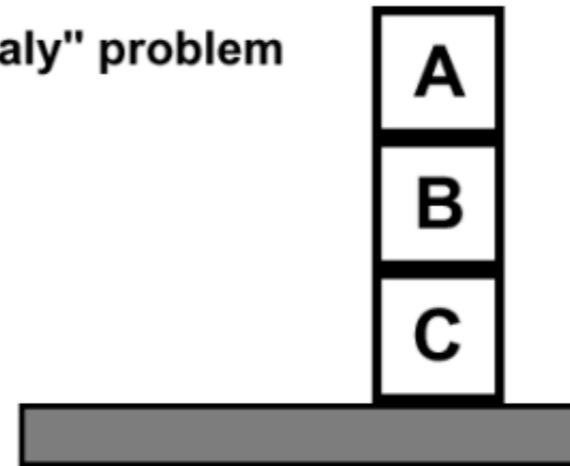


Start State

Clear(x) On(x,z) Clear(y)



~On(x,z) ~Clear(y)
Clear(z) On(x,y)



Goal State

Clear(x) On(x,z)

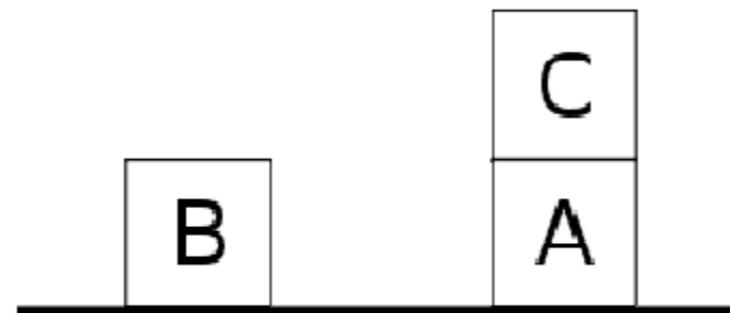


~On(x,z) Clear(z) On(x, Table)

on(c, table).
on(a, b).
on(b, c).
free(a).



Sussman Anomaly



- The Sussman Anomaly shows the limitations of non-interleaved planning methods
- Before this was described, people used to do planning by considering different subgoals in SEQUENCE
- The Anomaly will show that naively pursuing one subgoal X after you satisfy the other subgoal Y may not work because steps required to accomplish X might undo things subgoal Y

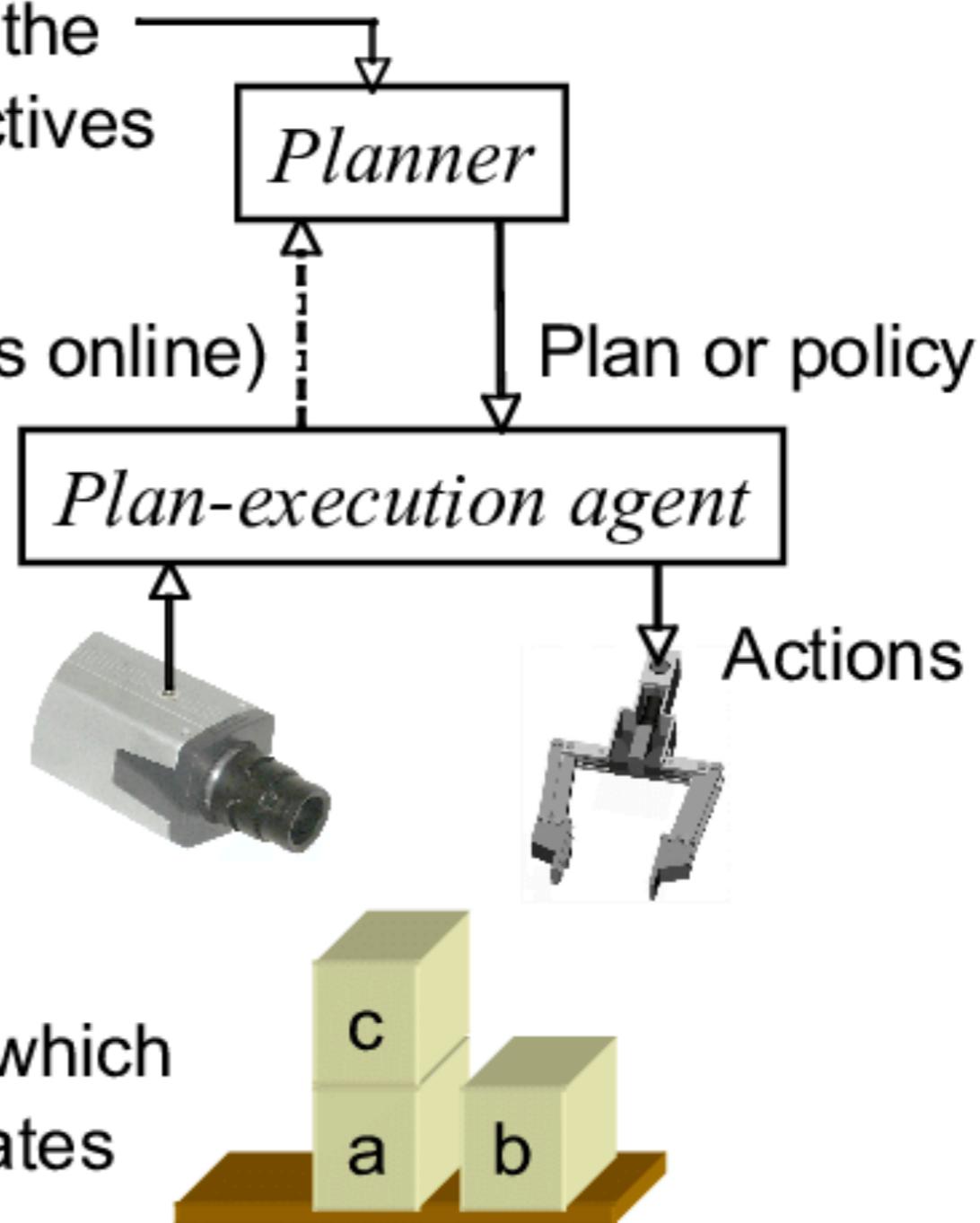


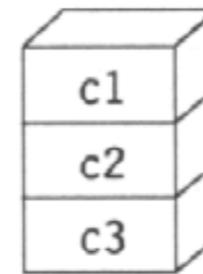
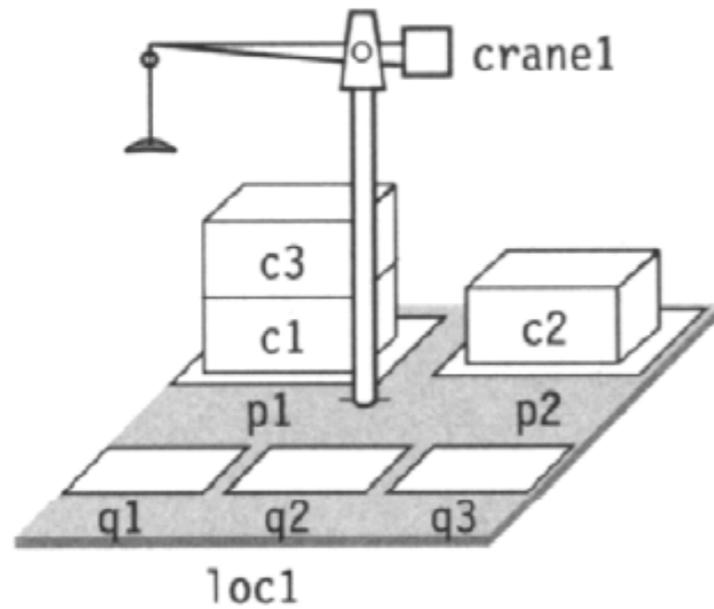
Descriptions of the world Σ , the initial state(s), and the objectives

Execution status (if planning is online)

Observations

The world Σ in which the agent operates





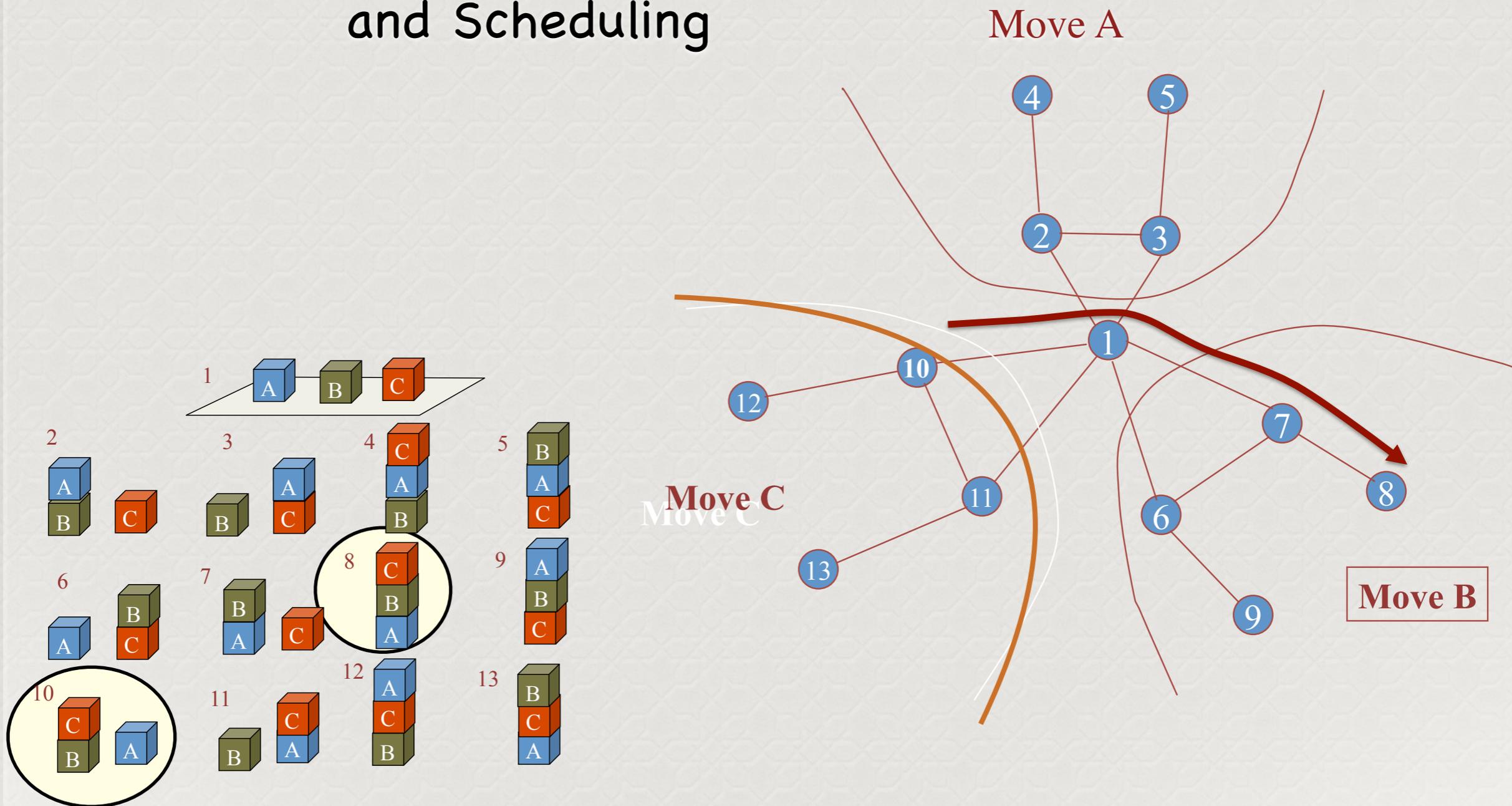
$s_0 = \{in(c3,p1), top(c3,p1), in(c1,p1), on(c3,c1), on(c1,pallet), in(c2,p2), top(c2,p2), on(c2,pallet), top(pallet,q1), top(pallet,q2), top(pallet,q3), empty(crane1)\}$

$g = \{on(c1,c2), on(c2,c3)\}$





KEPS-Knowledge Engineering for Planning and Scheduling





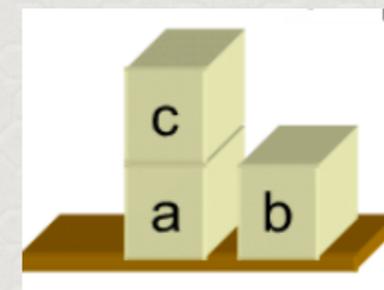
Exercício programa (Learn by doing)

Planner



Execution agent

Execution Plant





O estado da planta de execução consiste de uma lista de proposições lógicas descrevendo a configuração dos 3 elementos (blocos).



$[on(a, b), on(b, mesa), on(c, mesa), livre(a), livre(c)]$

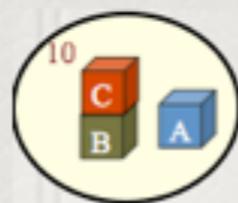


A dinâmica da planta de execução é dada por um único operador, $move(X, Y)$ que move o elemento X para cima do elemento Y .

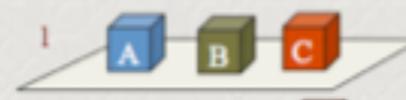
$move(c, mesa)$.

$remove([sobre(c, b)]I, s)$

$add([sobre(c, mesa), livre(b)]I, s)$



sobre(c, b)
sobre(b, mesa)
sobre(a, mesa)
livre(c)
livre(a)



sobre(c, mesa)
sobre(b, mesa)
sobre(a, mesa)
livre(c)
livre(b)
livre(a)



Existem várias implementações do mundo de blocos na internet e até no SWISH Prolog (lembrando que os exercícios feitos no SWISH são em princípio públicos). A exercício consiste portanto da escolha e modificação de um programa que não seja “naíve”, isto é, que não incorra na anomalia de Sussman.



O mundo de blocos - versão D-Lab

The screenshot shows the SWISH Prolog IDE interface. The main editor displays the following Prolog code:

```
1 go(Start, Goal) :-
2   bloco_fora_lugar(Start,Goal,H), %A função bloco_fora_lugar é a heurística
3   adiciona([[Start],H], [], Been_list), %Adiciona a configuração inicial à Been_list, que é
4   %uma lista contendo todos os nós que já foram expandidos e seus respectivos caminhos.
5   path(Start,Goal,Been_list).
6
7
8
9
10
11
12
13
14 path(Start,Goal, Been_list):- %Nessa chamada de path, ocorre a expansão dos nós
15   primeiro_lista_lista(Estado_Atual,Been_list),%Encontra o nó atual do primeiro elemento da li
16   solucoes(Estado_Atual,Solucao), %Encontra os possiveis candidatos para serem expandidos
17   primeiro_lista(Caminho,Been_list), %Retorna o caminho até o nó que está sendo expandido atua
18   del_first(Been_list,Been_list1), %Deleta o nó pai cujo filho será gerado a seguir
19   expande_no(Solucao,Goal,Caminho,Been_list1,New_Been_list), %Função que escolhe qual é o filh
20   path(Start,Goal,New_Been_list).%Chama a função de novo e continua verificando os estados até
21
22
23 ***** Funções Auxiliares *****
24
25 %Função que movimenta um bloco X para cima de um bloco Y
26 move_b(State,Next) :-
27   encontra_topo_lista(Y,State),
28   encontra_topo_lista(X,State),
29   atom_string(X,_),
30   atom_string(Y,_),
31   X \== Y,
32   deleta_lista(X,State,Next1),
33   insere_lista(Y,X,Next1,Next)
```

The interface includes a menu bar (File, Edit, Examples, Help), a search bar, and a status bar at the bottom showing the URL: <https://reinaldoswi.dlab-usp.org/p/Mundo de blocos - Rework 2.0.pl#tabbed-tab-0>. A query input field is visible on the right side of the editor.



O mundo de blocos - versão D-Lab

```
SWISH File Edit Examples Help 1 users online Search Q
```

```
1 go(Star
2 blo
3 adi
4 pat
5
6 path(
7 pri
8 Obj
9 pri
10 inv
11 wri
12 !.
13
14 path(St
15 pri
16 sol
17 pri
18 del
19 exp
20 pat
21
22
23 *****
24
25 %Função
26 move_b(
27 enc
28 enc
29 ato
30 ato
31 X \
32 del
33 ine
34 https://reinaldos
```

```
6
7 path(_,Goal, Been_List):- %Nessa chamada de path, considera-se o caso em que o objetivo foi
8 %encontrado
9 primeiro_lista_lista(Objetivo,Been_List),
10 Objetivo = Goal,
11 primeiro_lista(Caminho,Been_List),
12 inverte(Caminho,Caminho_I),
13 write(Caminho_I),
14 !.
15
16 path(Start,Goal, Been_list):- %Nessa chamada de path, ocorre a expansão dos nós
17 primeiro_lista_lista(Estado_Atual,Been_list),%Encontra o nó atual do primeiro elemento da
18 %lista
19 solucoes(Estado_Atual,Solucao), %Encontra os possiveis candidatos para serem expandidos
20 primeiro_lista(Caminho,Been_list), %Retorna o caminho até o nó que está sendo expandido
21 %atualmente
22 del_first(Been_list,Been_list1), %Deleta o nó pai cujo filho será gerado a seguir
23 expande_no(Solucao,Goal,Caminho,Been_list1,New_Been_list), %Função que escolhe qual é o
24 %filho mais promissor e coloca ele junto com seu caminho na lista contendo todos os nós
25 path(Start,Goal,New_Been_list).%Chama a função de novo e continua verificando os estados
%até chegar ao objetivo
```



```
30
31 %Função que movimenta um bloco X para cima de um bloco Y
32 move_b(State,Next) :-
33     encontra_topo_lista(Y,State),
34     encontra_topo_lista(X,State),
35     atom_string(X,_),
36     atom_string(Y,_),
37     X \== Y,
38     deleta_lista(X,State,Next1),
39     insere_lista(X,Y,Next1,Next).
40
41 %Função que movimenta um bloco X para a mesa
42 move_table_b(State,Next) :-
43     encontra_topo_lista(X,State),
44     atom_string(X,_),
45     vazia_lista(State),
46     deleta_lista(X,State,Next1),
47     insere_vazia_lista(X,Next1,Next),
48     Next \= State.
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```



Mundo

1 go(S

2

3

4

5

6 path

7

8

9

10

11

12

13

14 path

15

16

17

18

19

20

21

22

23 %%%%

24

25 %Fur

26 move

27

28

29 atom_string(X,_),

30 atom_string(Y,_),

31 X \== Y,

32 deleta_lista(X,State,Next1),

33 insere_lista(X,Y,Next1,Next)

34

35

36

37

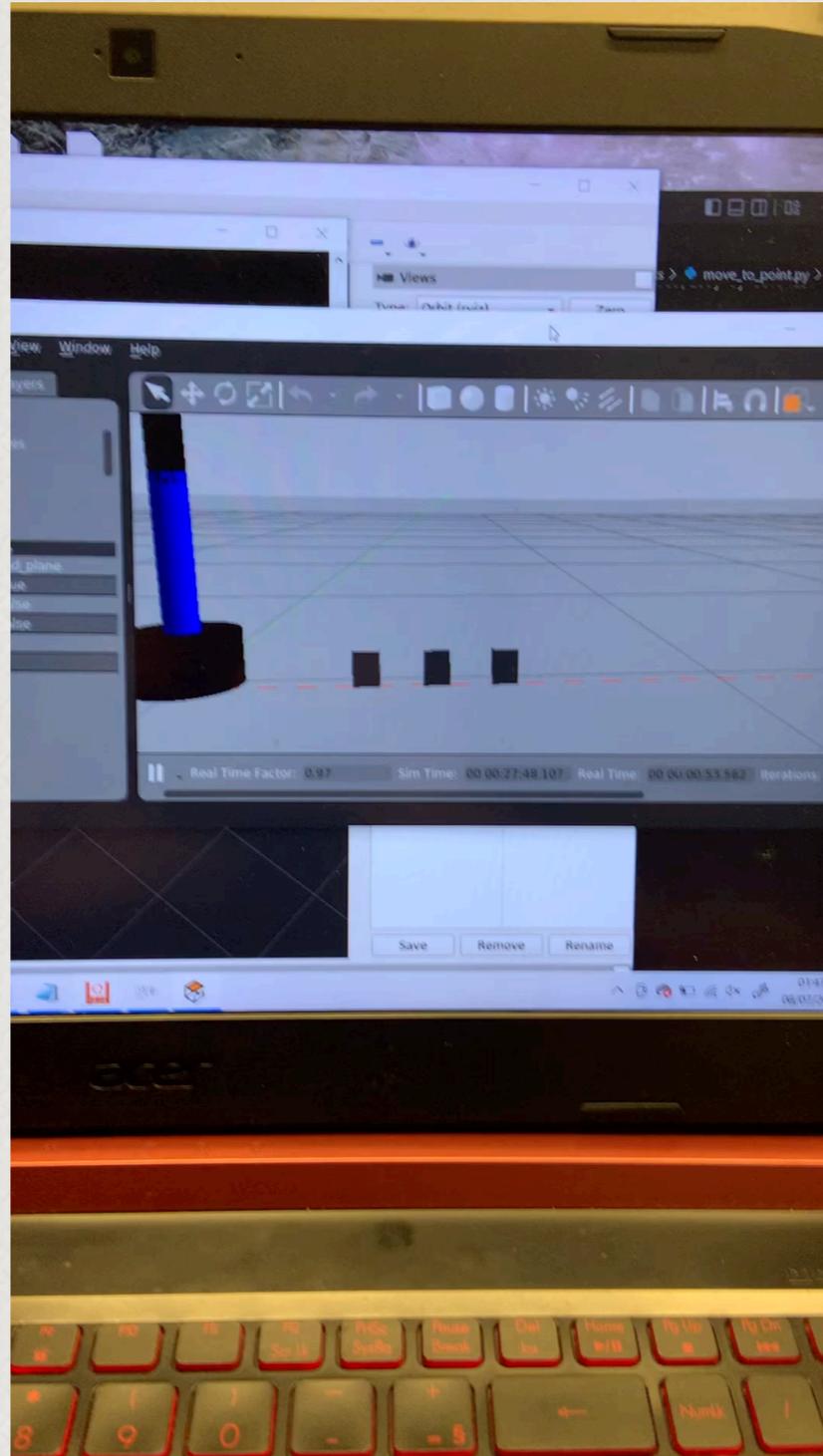
38

39

<https://reinaldoswi.dlab-usp.org/p/Mundo de blocos - Rework 2.0.pl#tabbed-tab-0>

Examples History Solutions

table results **Run!**





Na próxima aula continuaremos a discussão sobre estados preferenciais em um processo de planning e possíveis heurísticas para melhorar a performance dos planejadores.



Perguntas?